Optimal Link Scheduling for Delay-constrained Periodic Traffic over Unreliable Wireless Links

Yan Li, Haibo Zhang, Zhiyi Huang and Michael Albert
Department of Computer Science
University of Otago, Dunedin, New Zealand
{yanli, haibo, hzy, malbert}@cs.otago.ac.nz

***Abstract*—This paper investigates the problem of scheduling delay-constrained traffic in a single-hop wireless industrial network in which different source devices have different data rates. We aim to maximize the packet delivery reliability while meeting the deadline for each packet. The transmission scheduling problem is decomposed into two sub-problems: subperiod-based slot allocation and slot-based transmission scheduling. The former sub-problem is formulated as a nonlinear integer programming problem, and we present a solution with polynomial-time complexity by converting it to a linear integer programming problem. For the latter sub-problem, we demonstrate that the existence of a feasible optimal schedule depends on the order of the elements in the slot allocation vector produced by solving the former sub-problem. An algorithm is designed to compute a feasible slot allocation that sustains a realizable schedule. Simulation results demonstrate that our scheme ensures each device has almost the same packet delivery rate in different report periods, which is important for maintaining the stability of control systems.**

I. Introduction

Industrial automation using wireless sensor-actuator networks (WSANs) has attracted much attention from both the research and the industry sectors in recent years due to the significant advantages in terms of reducing deployment cost and providing high system-level flexibility. To encourage the uptake of WSANs in industry, three international standards, WirelessHART [7], ISA 100.11a [10] and IEEE 802.15.4e [1], have been released. All these standards use low-power radios compliant with the IEEE 802.15.4-2006 standard, which supports 16 channels in the 2.4GHz ISM band. Unfortunately, devices operating on these low-power radios are highly unreliable and link qualities are generally time-variant, especially in harsh environments such as factory floors with strong noise and frequent signal reflection from moving objects. Hence, a key challenge in the design of wireless industrial control networks is to provide ultra-reliable and real-time data communications despite the presence of channel loss.

A WSAN for industrial control is a distributed system of sensors, controllers and actuators interconnected over wireless links. Sensors periodically report measurements to controllers in which commands are generated based on the sensor data and in turn disseminated to actuators for control. The packets generated by the sensors commonly have hard deadlines, and the failure to deliver the packets before their deadlines can deteriorate the control performance and cause instability in the control system. To meet the stringent requirements on reliability and real-timeliness on wireless communications, all the recent industrial wireless standards [7] [10] [1] adopt the Time Division Multiple Access (TDMA) scheme and combine it with a deadline-constrained transmission scheduling policy. To enhance transmission reliability over lossy wireless channels lost packets have to be retransmitted which is commonly accomplished by reserving extra slots for retransmissions in TDMA-based schemes. A key question is how to allocate the extra slots and schedule retransmissions to maximize the probability for the controller to receive all packets within their deadlines. Even though many TDMA-based link scheduling schemes have been proposed for delay-optimal data gathering in wireless sensor networks [5][2][12], most schemes assume that all packets generated in each data gathering round have the same deadline. Hence, these schemes are not suitable to networks with heterogenous traffic rates, as commonly arises in industrial applications.

In this paper we investigate the problem of scheduling periodic heterogenous traffic with delay constraints in single-hop wireless industrial control networks based on the following general conditions: different sensor devices can have different packet generation rates, and different wireless links can have different packet loss rates. Our objective is to optimally allocate time slots and schedule packet transmissions so as to maximize the rewards on packet reception. Specifically, this work has the following major contributions:

- We present a theoretical model for scheduling delay-constrained heterogeneous traffic in single-hop wireless networks, and propose a two-stage scheduling approach: subperiod-based slot allocation and slot-based transmission scheduling.
- We formulate the subperiod-based slot allocation problem as a nonlinear integer programming optimization problem, and present an approach to convert it to a linear integer programming problem. A polynomial-time algorithm is designed to compute the optimal solution for the subperiod-based slot allocation problem.
- The solution for the subperiod-based slot allocation problem produces a slot allocation vector that specifies how many slots are allocated to which report period of which device. While the optimal reward depends only on the values of the elements in the allocation vector, we demonstrate that the existence of a realizable optimal schedule

978-1-4799-3360-0/14/$31.00

depends on the order of the elements in the allocation vector. An algorithm is designed to compute a feasible order that represents a realizable schedule.
- We evaluate the performance of our scheme through simulations in comparison with existing solutions. Simulation results demonstrate that our scheme achieves much higher packet delivery rate, and also provides fairness on packet delivery rate among different subperiods of each device.

The rest of this paper is organized as follows: Section II summarizes the related work. Section III gives the system model and problem formulation. Section IV and Section V present the optimal solutions for the subperiod-based slot allocation problem and the slot-based transmission scheduling problem, respectively. Section VI presents the simulation results. Finally, the paper is concluded in Section VII.

II. Related Work

The problem of scheduling delay-constrained traffic over lossy wireless networks has received intensive interest in recent years. Willig [15] discussed the challenges faced by wireless industrial networks and gave a survey on the promising technologies for providing reliable and real-time communication in wireless industrial networks. TDMA-based link scheduling for data collection in wireless sensor networks has also been well studied [16] [13] [2]. However, most of the existing solutions can only deal with homogeneous traffic in which all packets generated in the same data collection period have the same deadline. Existing scheduling schemes for cellular networks [6] and wireless LAN [4] [14] either consider only soft deadlines or assume constant-bit-rate traffic, and are thus not suitable for reliability- and delay-sensitive wireless industrial control networks. Saifullah et.al [11] presented a general model for link scheduling in WirelessHART networks, and proved that the real-time scheduling problem is NP-complete. Hou and Kumar gave a survey of the recent results on real-time wireless networking in [9].

The related work in the literature also includes extensive studies on task scheduling for single processor systems, in which each task is characterized by an arrival time, a demand on CPU time, and a deadline. As opposed to the traditional task scheduling problem, the number of time slots allocated for transmitting a packet (which corresponds to the amount of CPU time for executing a task) in our problem is not known, and we aim to optimize the slot allocation to maximize the reward (i.e. reliability). Thus existing scheduling policies such as EDF cannot be applied directly to solve our problem. In [3] the optimal reward-based scheduling problem was investigated for periodic real-time tasks based on the *imprecise computation* model, in which each task is composed of a mandatory part and an optimal part. The former must be guaranteed to be completed before the deadline. The longer the optional part is executed before the deadline, the higher the reward. The authors proved that, for linear and concave reward functions, an optimal schedule exists when the optional service time for a task is constant at every instance, but the problem is NP-hard for convex reward functions.

Another closely related work is [8], in which the authors studied the problem of scheduling periodic real-time task in which different tasks can have different reward requirements. The necessary and sufficient condition for scheduling feasibility is established and a greedy online scheduling policy, called *Greedy Maximizer*, is proposed. However, the greedy scheme is feasibility optimal only when the periods of all tasks are the same. The authors also proved that the Greedy Maximizer policy is a 2-approximation policy. Another major difference is that the solution presented in [8] is based on *long-term average reward*, and the same task in different periods may produce different rewards. In our solution there is at most one device that can have different number of allocated slots in different periods, and the maximum difference is one slot. The performance of our solution is compared with Greedy Maximizer using simulations.

III. System Model and Problem Formulation

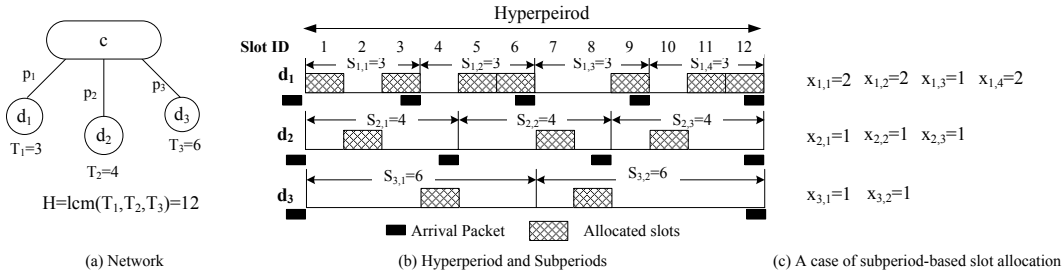
A. System Model

Consider a wireless industrial network with one controller c, and N wireless sensor devices numbered $d_1, d_2, \cdots, d_N$. Each device is equipped with a single half-duplex radio transceiver, implying that the controller cannot receive packets from more than one sensor device at the same time. All sensor devices can directly communicate with the controller (i.e. single hop). Time is synchronized and divided into slots of equal length, with each allowing the transmission of one data packet and its associated acknowledgement. It is assumed that the packet loss rates for different wireless links are independent and follow the Bernoulli model. For each individual packet transmission from d_i to c, the packet is lost with probability p_i. Note that, in our model, a packet transmission is successful if and only if the transmitter has received the acknowledgement for that packet. Hence the packet loss probability for each link takes into account the losses of both the data packet and the acknowledgement. The above packet loss model is in compliance with the WirelessHART standard which requires link scheduling to be performed centrally at the network manager based on the long-term channel outage probability.

Each sensor device d_i reports periodically to the controller with a fixed report interval of T_i slots. Each periodic traffic contains only one data packet that is generated just before the start of the corresponding report period. The deadline for a packet generated by d_i coincides with its period T_i. If the packet is not successfully delivered to the controller within its deadline, it will be dropped as the next report period starts.

B. Problem Formulation

We first define two important terms: **subperiod** and **hyperperiod**, which will be used throughput the paper. A subperiod for d_i is simply a report period with fixed length of T_i slots, and we use $S_{i,m}$ to denote the mth subperiod of device d_i. A hyperperiod is defined as a period with fixed length of H slots, where H is the least common multiple of all T_i, i.e. $H = \text{lcm}\{T_1, \cdots, T_N\}$. Thus, for device d_i, a hyperperiod contains $\frac{H}{T_i}$ subperiods with each having a packet



(a) Network (b) Hyperperiod and Subperiods (c) A case of subperiod-based slot allocation

Fig. 1. An illustrative example

to be transmitted to the controller. Figure 1 gives an example illustrating these two terms. It can be seen that, by introducing these two concepts, we only need to study the problem of scheduling packet transmissions for one hyperperiod since the other hyperperiods just repeat the same schedule.

To enhance packet transmission reliability, the widely adopted approach based on TDMA is to reserve multiple slots for retransmission. Let $x_{i,m}$ be the number of slots allocated for transmitting the packet generated for $S_{i,m}$. Obviously, the packet fails to reach the controller only when all transmissions in the $x_{i,m}$ allocated slots fail, and the corresponding failure probability is $p^{x_{i,m}}$. Therefore, the success probability for the packet to reach the controller using $x_{i,m}$ slots is $1-p_i^{x_{i,m}}$. It can be seen that the success probability depends on only $x_{i,m}$, rather than where these $x_{i,m}$ slots are allocated.

To construct the transmission schedule, the number of time slots allocated to each arrival packet in a hyperperiod needs to be determined. However, slot allocations to different packets are strongly correlated. The more slots allocated to one packet, the higher success probability the packet has, but the fewer slots remain for other packets. In this work we aim to achieve optimal slot allocation by maximizing the following objective function:

$$\mathcal{R}=\sum_{i=1}^{N}\sum_{m=1}^{H/T_i} w_i(1-p_i^{x_{i,m}}), \tag{1}$$

where w_i represents the weight assigned to the packets generated by d_i. In some applications, the packets generated by some devices are more important than others, and need to get higher reliability of delivery. This can be achieved by properly configuring the weight for each device. For the case where all packets have the same weight, $\mathcal{R}$ corresponds to the expected number of packets received by the controller in one hyperperiod.

Since all packet transmissions must be scheduled within the hyperperiod, the first constraint on our problem is that the total number of allocated slots should not exceed the length of the hyperperiod, that is,

$$\sum_{i=1}^{N}\sum_{m=1}^{H/T_i} x_{i,m}=H. \tag{2}$$

Each packet generated by d_i has a deadline that coincides with its period T_i, which enforces that the number of time slots allocated to each packet of d_i must not exceed T_i, which is expressed as,

$$x_{i,m}\le T_i,\quad \forall i\in[1,N],\forall m\in[1,H/T_i]. \tag{3}$$

We formulate the optimal slot allocation problem as the following optimization problem:

$$\begin{array}{ll}\text{maximize} & \mathcal{R}=\sum_{i=1}^{N}\sum_{m=1}^{H/T_i} w_i(1-p_i^{x_{i,m}})\\ \text{such that} & (2)\text{ and }(3)\text{ are satisfied.}\end{array} \tag{4}$$

Note that the solution of the above optimization problem is a vector $\vec{x}=[x_{i,m}]$, which is not a real transmission schedule. We refer to the above optimization problem as the **subperiod-based slot allocation problem**, and solve it in Section IV. Once getting the optimal $x_{i,m}$ for each d_i in its mth subperiod, we need to compute the positions for these $x_{i,m}$ slots in the corresponding subperiods. We refer to this problem as the **slot-based transmission scheduling problem** and solve it in Section V. For convenience of reading, a summary of the notations used in this section is given in Table I.

TABLE I
SUMMARY OF THE MAIN NOTATIONS

Symbol	Meaning
N	the number of sensor devices
d_i	the ith sensor device
c	the controller
T_i	the fixed report period of d_i
H	the length of the hyperperiod
$S_{i,m}$	the mth subperiod of the ith sensor device
$x_{i,m}$	the number of slots assigned to d_i in $S_{i,m}$
p_i	packet loss probability for link between d_i and c

IV. SUBPERIOD-BASED ALLOCATION

The subperiod-based allocation problem formulated in Section III is a non-linear integer programming problem, and such problems may be NP-hard. In this section, we first convert this problem into a linear integer programming problem, and then present a polynomial-time algorithm to compute the optimal solution for this problem.

A. Non-linear Problem to Linear Problem Conversion

Our conversion approach is based on mulitset theory. As in a set the order of elements in a multiset is not important but,

unlike a set, duplicate elements are permitted. For example, $\{a, a, b, b, b\}$ is a multiset whose underlying set is $\{a, b\}$. The multiplicity of a member of a multiset is the number of times it occurs, and the total number of elements in a multiset (allowing for repetition) is called the cardinality of the multiset. In the foregoing example, the multiplicities of a and b are respectively 2 and 3, and the cardinality of the multiset is 5. So this multiset can also be expressed in a more general way as $\{2 \cdot a, 3 \cdot b\}$.

In Eqn. (1), $(1 - p_i^{x_{i,m}})$ is the probability that device d_i successfully transmits a packet to the controller in $x_{i,m}$ time slots. It is equal to $\sum_{k=1}^{x_{i,m}} (1 - p_i) p_i^{k-1}$ where $(1 - p_i) p_i^{k-1}$ is the probability that the packet transmission fails in the first $k - 1$ time slots and succeeds in the k^{th} time slot. Thus, $\mathcal{R}$ in Eqn. (1) can be rewritten as follows:

$$\mathcal{R} = \sum_{i=1}^{N} \sum_{m=1}^{H/T_i} \sum_{k=1}^{x_{i,m}} w_i (1 - p_i) p_i^{k-1} \tag{5}$$

Let $\mathcal{C}$ be a multiset defined as follows:

$$\mathcal{C} = \Big\{ b_{i,k} = w_i \left(1 - p_i\right) p_i^{k-1} \mid i \in [1, N], m \in [1, \tfrac{H}{T_i}], k \in [1, x_{i,m}] \Big\}.$$

Obviously, $\mathcal{R}$ can be expressed as the sum of all elements in $\mathcal{C}$. According to constraint (2), the cardinality of $\mathcal{C}$ must be H. We define another multiset $\mathcal{B}$ as follows:

$$\mathcal{B} = \left\{ \frac{H}{T_i} \cdot b_{i,k} \mid b_{i,k} = w_i \left(1 - p_i\right) p_i^{k-1}, i \in [1, N], k \in [1, T_i] \right\}$$

Actually, $\mathcal{B}$ is a special case of mutlitset $\mathcal{C}$ where $x_{i,m} = T_i$. The following lists all the elements in $\mathcal{B}$ for the example given in Fig. 1 (a) where $w_1 = w_2 = w_3 = 1$.

$$\begin{array}{lll} 4 \cdot (1 - p_1) & 3 \cdot (1 - p_2) & 2 \cdot (1 - p_3) \\ 4 \cdot (1 - p_1) p_1 & 3 \cdot (1 - p_2) p_2 & 2 \cdot (1 - p_3) p_3 \\ 4 \cdot (1 - p_1) p_1^2 & 3 \cdot (1 - p_2) p_2^2 & 2 \cdot (1 - p_3) p_3^2 \\ & 3 \cdot (1 - p_2) p_2^3 & 2 \cdot (1 - p_3) p_3^3 \\ & & 2 \cdot (1 - p_3) p_3^4 \\ & & 2 \cdot (1 - p_3) p_3^5 \end{array}$$

Due to the constraints given in (2) and (3), the multiset $\mathcal{C}$ for any feasible solution of the subperiod scheduling problem must be a subset of $\mathcal{B}$. For example, the multiset $\mathcal{C}$ for the solution given in Fig. 1 (c) is $\{4 \cdot (1 - p_1),\ 3 \cdot (1 - p_2),\ 2 \cdot (1 - p_3),\ 3 \cdot (1 - p_1) p_1\}$. In the following we prove that the subperiod-based scheduling problem can be converted into a linear integer programming problem.

Theorem 1. *The maximum $\bar{\mathcal{R}}$ obtained by solving the following integer programming problem must be the maximum value for $\mathcal{R}$ obtained by solving the non-linear integer programming problem given in (4).*

$$\textit{maximize} \qquad \bar{\mathcal{R}} = \sum_{i=1}^{N} \sum_{k=1}^{T_i} y_{i,k} w_i (1 - p_i) p_i^{k-1} \tag{6}$$

$$\textit{s.t.} \qquad \sum_{i=1}^{N} \sum_{k=1}^{T_i} y_{i,k} = H \tag{7}$$

$$0 \le y_{i,k} \le \tfrac{H}{T_i}. \quad \forall i \in [1, N], \forall k \in [1, T_i] \tag{8}$$

$$y_{i,k} \ge y_{i,k+1} \quad \forall i \in [1, N], \forall k \in [1, T_i - 1] \tag{9}$$

Proof: Similar to $\mathcal{R}$, $\bar{\mathcal{R}}$ can also be expressed as the sum of elements in a multiset $\bar{\mathcal{C}}$ defined as follows:

$$\bar{\mathcal{C}} = \left\{ y_{i,k} \cdot b_{i,k} \mid b_{i,k} = w_i \left(1 - p_i\right) p_i^{k-1}, i \in [1, N], k \in [1, T_i] \right\}$$

Constraint (7) ensures that the cardinality of $\bar{\mathcal{C}}$ must be H, which corresponds to constraint (2). Constraint (8) guarantees that no subperiod of d_i is allocated more than T_i time slots, which corresponds to constraint (3). The three constraints of (7), (8), and (9) also guarantee that any solution of the converted linear integer programing problem (i.e. $\vec{y} = [y_{i,k}]$) can be mapped onto a feasible solution of the problem formulated in (4) (i.e. $\vec{x} = [x_{i,m}]$), and vice versa (this will be proved in Lemma 2 in Section IV-B). Hence, the maximum value for $\bar{\mathcal{R}}$ must be the maximum value for $\mathcal{R}$. □

B. Optimal Solution

Let $\mathcal{O}$ be a multiset that contains the H largest elements in $\mathcal{B}$. We have the following Lemma 1.

Lemma 1. *The sum of all elements in $\mathcal{O}$ must be the maximum value for $\bar{\mathcal{R}}$.*

Proof: According to the definition of $\mathcal{O}$, this lemma must hold if the multipicities of elements in $\mathcal{O}$ satisfy constraints (7), (8) and (9). Constraints (7) has already been satisfied because the cardinality of $\mathcal{O}$ is H. Constraint (8) is obviously satisfied according to the definition of $\mathcal{B}$. Now we prove constraint (9) is satisfied. Since $0 \le p_i < 1$ for any $i \in [1, N]$, $b_{i,k-1} > b_{i,k}$. If there is a $b_{i,k}$ in $\mathcal{O}$, there must be a corresponding $b_{i,k-1}$ in $\mathcal{O}$. Hence, $y_{i,k} \ge y_{i,k+1}$. □

Therefore, to compute the optimal solution for the converted linear integer programming problem, we only need to calculate the multiplicity $y_{i,k}$ for each $b_{i,k}$ in $\mathcal{O}$.

Property 1. *For any two elements $b_{i,k}$ and $b_{m,n}$ in $\mathcal{O}$ where $b_{i,k} > b_{m,n}$, we have $y_{i,k} = \frac{H}{T_i}$ and $y_{m,n} \le \frac{H}{T_m}$.*

Since $\mathcal{O}$ is a subset of $\mathcal{B}$, $b_{i,k}$ and $b_{m,n}$ are also in $\mathcal{B}$ in which the multiplicities for $b_{i,k}$ and $b_{m,n}$ are $\frac{H}{T_i}$ and $\frac{H}{T_m}$, respectively. Since $b_{m,n}$ is in $\mathcal{O}$ and $b_{i,k} > b_{m,n}$, according to the definition of $\mathcal{O}$, all elements that are equal to $b_{i,k}$ must be in $\mathcal{O}$. Hence, $y_{i,k} = \frac{H}{T_i}$. Similarly, if there is another element in $\mathcal{O}$ that is smaller than $b_{m,n}$, $y_{m,n}$ must be $\frac{H}{T_m}$; otherwise, $b_{m,n}$ can be partly in $\mathcal{O}$ and partly in $\mathcal{B} \backslash \mathcal{O}$ due to the constraint on the cardinality of $\mathcal{O}$, in which case $y_{m,n} < \frac{H}{T_m}$. Hence, $y_{m,n} \le \frac{H}{T_m}$.

Based on Property (1), $y_{i,k}$ for $b_{i,k}$ can be computed as follows:

$$y_{i,k} = \begin{cases} \dfrac{H}{T_i}, & \text{if } b_{i,k} \in \mathcal{O} \text{ and } b_{i,k} \notin \mathcal{B} \setminus \mathcal{O} \\ H - \displaystyle\sum_{b_{m,n} \in \mathcal{O} \text{ and } b_{m,n} \notin \mathcal{B} \setminus \mathcal{O}} y_{m,n}, & \text{otherwise.} \end{cases} \tag{10}$$

Once we get the multiplicity of each element in $\mathcal{O}$, the optimal solution for the problem in (4) can be obtained by mapping $\vec{y}_{i,k}$ to $\vec{x}_{i,m}$ based on the following lemma.

Lemma 2. *Any solution $\vec{y}$ for the converted linear integer programming problem can be mapped to a solution $\vec{x}$ for the non-linear integer programming problem given in (4), and vice versa.*

Proof: **$\vec{x}$ to $\vec{y}$ mapping**: $y_{i,k}$ is equivalent to the number of subperiods of d_i in which $x_{i,m} \geq k$.
$\vec{y}$ to $\vec{x}$ mapping: If $y_{i,k} = \frac{H}{T_i}$, then $y_{i,1} = y_{i,2} = \cdots = y_{i,k-1} = \frac{H}{T_i}$ according to Property 1. This means that each subperiod of d_i is allocated at least k time slots. If $b_{i,k+1}$ is not in $\mathcal{O}$, it means that no subperiod of d_i is allocated $k+1$ time slots. Thus $x_{i,1} = x_{i,x} = \cdots = x_{H/T_i} = k$ when $y_{i,k} = \frac{H}{T_i}$ and $b_{i,k+1}$ is not in $\mathcal{O}$. If $y_{i,k} = \frac{H}{T_i}$ and $0 < y_{i,k+1} < \frac{H}{T_i}$, it means at least k slots are assigned to each subperiod of d_i according to above discussion, and there are $y_{i,k+1}$ subperiods of d_i that can obtain one more slot for packet transmission. □

Algorithm 1 (denoted as OPT-SP) gives the steps of computing the optimal $\vec{x}_{i,m}$ for problem (4). The complexity of

Algorithm 1: OPT-SP

Input: $\vec{T} = [T_i]$, $\vec{p} = [p_i]$
Output: $\vec{x} = [x_{i,m}]$
1 $H = lcm\{T_1, T_2, \cdots, T_N\}$;
2 $x_{i,m} = 0$; $\forall i \in [1, N]$, $\forall m \in [1, \frac{H}{T_i}]$
3 Construct the underlying set for $\mathcal{B}$, denoted by $\mathcal{B}_u$;
4 Choose the H largest elements in $\mathcal{B}_u$ into a queue Q in a non-increasing order;
5 $b_{i,k} = Q \rightarrow next$;
6 **while** $H > 0$ **do**
7 **for** $m = 1, 2, \cdots, \frac{H}{T_i}$ **do**
8 Update $x_{i,m} = x_{i,m} + 1$;
9 $H = H - 1$;
10 **if** $H = 0$ **then**
11 Break;
12 $b_{i,k} = Q \rightarrow next$;
13 return $\vec{x}$;

Algorithm 1 is dominated by the operation of choosing the H largest elements from $\mathcal{B}_u$ and sorting them in a non-increasing order. Since the cardinality of $\mathcal{B}_u$ is $\sum_{i=1}^{N} T_i$. This can be efficiently solved using the order statistic algorithm with time complexity of $O(\sum_{i=1}^{N} T_i + H \log H)$.

Lemma 3. *There is at most one device the subperiods of which may be assigned different numbers of time slots and these assignments differ by at most one.*

Proof: The for-loop (line 7 to line 11) in Algorithm 1 guarantees that, once a $b_{i,k}$ is chosen, each subperiod of d_i will get a slot as long as there are available slots to allocate. So there is only one device the subperiods of which can get different numbers of time slots and this can arise only due to the insufficient allocation of the last round to the subperiods. Hence, the maximum difference between allocations to the subperiods is one time slot. □

The above lemma indicates that our solution provides roughly the same packet delivery rate in different subperiods for each device. However, the fairness among different devices cannot be guaranteed. In Section VI, we will demonstrate that relative fairness can be achieved by adjusting the weight assigned to each device.

V. Slot-based Scheduling

The optimal solution presented in previous section only gives the optimal number of time slots assigned to each subperiod, but does not specify how the assigned slots are allocated (i.e. which slots are assigned to which device in which subperiod). In this section we will first analyze the scheduling feasibility for a given slot allocation solution, and then present a solution to construct the optimal schedule.

A. Scheduling Feasibility

Since the radio channel between the controller and each device is shared, it means at most one device can communicate with the controller in any time slot. However, the subperiod-based slot allocation problem formulated in Section III-B does not take into account this constraint. Thus it is uncertain that the output of Algorithm 1 is realizable.

Lemma 4. *Constraints (2) and (3) are necessary but not sufficient for the existence of a feasible schedule.*

Proof: We define a decision variable $z_{i,j}$ as

$$z_{i,j} = \begin{cases} 1, & \text{if } d_i \text{ is scheduled in time slot } j \\ 0, & \text{if } d_i \text{ is not scheduled in time slot } j \end{cases}$$

We use $Z = \{z_{i,j}, \forall i \in [1, N], \forall j \in [1, H]\}$ to represent a transmission schedule for one hyperperiod. Obviously, Z is feasible if and only if the following constraint is satisfied.

$$\sum_{i=1}^{N} z_{i,j} = 1, \forall j \in [1, H], \tag{11}$$

which ensures that each slot can be assigned to only one sensor device. Eqn. (11) also implicitly ensures that the total number of slots assigned to all sensor devices cannot exceed H, that is,

$$\sum_{i=1}^{N} \sum_{k=1}^{H} z_{i,j} = H. \tag{12}$$

The number of time slots allocated to d_i in the mth subperiod is $x_{i,m}$, that is,

$$\sum_{j=(m-1)\cdot T_i+1}^{mT_i} z_{i,j} = x_{i,m}, \forall i \in [1, N], m \in [1, \frac{H}{T_i}] \tag{13}$$

Since $z_{i,j}$ is either 0 or 1, $x_{i,m} \leq T_i$. Hence constraint (3) must be satisfied. Based on Eqns. (12) and (13), we have

$$\sum_{i=1}^{N} \sum_{j=1}^{H} z_{i,j} = \sum_{i=1}^{N} \sum_{m=1}^{H/T_i} \sum_{j=(m-1)\cdot T_i+1}^{mT_i} z_{i,j} = \sum_{i=1}^{N} \sum_{m=1}^{H/T_i} x_{i,m} = H, \tag{14}$$

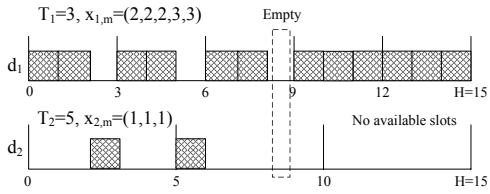


Fig. 2. Example for the Influence by the Order of $x_{i,m}$.

Thus constrains (2) and (3) are the necessary conditions for the existence of a feasible schedule.

However, the two constrains (2) and (3) are not sufficient for the existence of a feasible schedule as Eqn. (11) cannot be guaranteed. Fig. 2 give a counterexample in which the two constraints are satisfied, but no feasible schedule exists. The network consists of two devices d_1 and d_2 with $T_1 = 3$ and $T_2 = 5$. The number of slots allocated to the five subperiods of d_1 is 2, 2, 2, 3, and 3, respectively. Device d_2 is assigned with one slot for each of its subperiod. Since $x_{1,3} = x_{1,4} = 3$, d_1 must be scheduled in any slot from slot 10 to slot 15, which prevents d_2 from obtaining a slot in this period. Thus the subperiod-based slot allocation is not realizable. □

Let $\vec{x_i} = [x_{i,1}, ..., x_{i,m}]$ be a vector of the number of slots allocated to the subperiods of device d_i. It is worth noting that, in Problem (4), the optimal value for $\mathcal{R}$ depends only on the value of each $x_{i,m}$, and not on the order they appear in $\vec{x_i}$. Therefore, any permutation of $\vec{x_i}$ is still the optimal solution of Problem (4). In the example given in Fig. 2, a feasible schedule exists for a permutation $(3, 2, 3, 2, 3)$. In the following we show that, given any subperiod-based slot allocation computed by Algorithm 1, we can always find a realizable permutation of $\vec{x}$ to construct a feasible schedule.

B. Computation of a Realizable Permutation of $\bar{x}_i$

According to Lemma 3, there is only one sensor device which will be allocated different numbers of slots in different subperiods, and this device is represented by d_i^*. Hence, we only need to figure out a permutation for $\vec{x}_i^*$ that allows a feasible schedule. Let $\min(\vec{x}_i^*) = \min\{x_{i,1}^*, ..., x_{i,m}^*\}$. Suppose slot allocation for all devices except d_i^* is done, and each subperiod of d_i^* is assigned $\min(\vec{x}_i^*)$ time slots. Then the number of surplus slots that need to be further allocated is:

$$N_s = \sum_{m=1}^{H/T_i^*} x_{i,m}^* - \frac{H}{T_i^*} \min(\vec{x}_i^*). \quad (15)$$

According to Lemma 3, the number of slots allocated to different subperiods of d_i^* differs by at most one. Thus the N_s surplus slots need to be assigned to N_s subperiods of d_i^*, with each subperiod getting only one surplus slot.

Lemma 5. *There must be a feasible schedule policy that allows N_s subperiods of d_i^* to consume the N_s surplus slots, with each subperiod consuming only one surplus slot.*

Proof: We use a similar approach presented in [8] to prove this lemma based on the max-flow min-cut theorem for a flow network. We construct a flow network that has a source, a sink and three internal layers $L1$, $L2$ and $L3$. $L3$ contains H vertices, with each one representing a time slot. Each one is connected to the sink by an edge of capacity one, enforcing that each slot can only be allocated to one device. For each device d_i, there are H/T_i vertices in $L2$. The i^{th} vertex is connected by edges to the individual slots in the i^{th} subperiod for d_i, and each edge has infinite capacity. For each device d_i, there are T_i vertices in $L1$. Each vertex is connected to every vertex associated with d_i in $L2$ by an edge of capacity one. For each device d_i, there is an edge from the source to each vertex associated with d_i in $L1$, and the capacity of the edge to the k^{th} such vertex is the number of subperiods in which d_i is scheduled at least k times.

For such a flow network, the capacity of a bounded cut is at least the maximum of the capacities of the two trivial cuts (isolating the sink or source vertex), and both of these are H. According to the max-flow min-cut theorem, the maximum flow of this network is H. Since computing a maximum flow will also give a feasible schedule, there must be a feasible schedule policy to assign the N_s surplus slots for d_i^*. □

According to Lemma 5, computing a maximum flow gives a schedule and this can be done using Ford-Fulkerson algorithm effectively. However, the Ford-Fulkerson algorithm has high time complexity, $O(Ef)$, where E is the number of edges and f is the maximum flow in the network. In the following we present a solution based on scheduling capacity.

Let Δ_l be the number of surplus slots by the end of the lth subperiod of d_i^* after the mandatory slots are allocated. Then

$$\Delta_l = lT_i^* - \sum_{\substack{j=1 \\ j \neq i}}^{N} \lfloor \frac{lT_i^* x_j}{T_j} \rfloor - l \cdot \min(\vec{x}_i^*), \forall l = [1, \frac{H}{T_i^*}] \quad (16)$$

where $\sum_{\substack{j=1 \\ \neq i}}^{N} \lfloor \frac{lT_i^* x_j}{T_j} \rfloor$ is the number of slots that must be assigned to other devices except d_i^* by the end of the lth subperiod (all other devices have the same number of slots allocated to each subperiod, which is denoted by x_j for d_j). $l \cdot min(x_{i,m}^*)$ is the number of time slots (excluding the surplus slots) allocated to d_i^* by the end of the lth subperiod. Obviously, $\Delta_{H/T_i^*} = N_s$. However, Δ_l only gives the maximum available surplus slots by the end of the lth subperiod. The actual usable surplus slots also depends on the Δ_ls of the subperiods after the lth subperiod. That is, the actual number of usable surplus slots assigned by the end of the lth subperiod should be no larger than $\min\{\Delta_{l+1}, \Delta_{l+2}, ..., \Delta_{H/T_i^*}\}$. We use the following example to illustrate it. Suppose three devices d_1, d_2, and d_3 have subperiods $T_1 = 3, T_2 = 9$ and $T_3 = 27$. After Algorithm 1, we could generate $\vec{x_1} = [3, 3, 2, 2, 2, 2, 2, 2, 2]$, $\vec{x_2} = [2, 2, 2]$, and $\vec{x_3} = [1]$. According to Equation (16), we have $\vec{\Delta} = [1, 2, 1, 2, 3, 2, 3, 4, 2]$ for the subperiods of d_1. Even though $\Delta_2 = 2$, we cannot assign 2 surplus slots by the end of the second subperiod because $\Delta_3 = 1$ is smaller than 2. That means, the maximum available surplus slots by the end of the

3rd subperiod is just 1, therefore, by the end of the second subperiod at most 1 surplus slot could be assigned. Based on this observation, we divide the subperiods for d_i^* into different groups using the following policy: let $\Delta_{min} = \Delta_{H/T_i^{(*)}}$. The Δs are scanned in an order from Δ_{H/T_i^*} to Δ_1. If $\Delta_l \geq \Delta_{min}$, we include Δ_l in the group that Δ_{min} belongs to; otherwise $\Delta_{min} = \Delta_l$ and we put Δ_l in a new group (line 4 to line 10 in Algorithm 2). For the aforementioned example, the Δs for the subperiods of d_1 can be divided into the following 2 groups: $(1, 2, 1)$ and $(2, 3, 2, 3, 4, 2)$. Let N_g denote the number of groups and $Ps[j]$ represent the index of the last subperiod in the j^{th} group.

Lemma 6. *If group j consumes $\Delta_{Ps[j]} - \Delta_{Ps[j-1]}$ ($1 \leq j \leq N_g$, and $\Delta_{Ps[0]} = 0$) surplus slots, the sum of consumed surplus slots for all the N_g groups is N_s.*

Proof: The sum of consumed surplus slots in the N_g groups should be $\Delta_{Ps[1]} + (\Delta_{Ps[2]} - \Delta_{Ps[1]}) + \cdots + \Delta_{Ps[N_g]-1} - \Delta_{Ps[N_g]-2} + \Delta_{Ps[N_g]} - \Delta_{Ps[N_g]-1} = \Delta_{Ps[N_g]} = \Delta_{H/T_i^*} = N_s$. □

Lemma 7. *The kth group contains at least $\Delta_{Ps[k]} - \Delta_{Ps[k-1]}$ number of subperiods; Any $\Delta_{Ps[k]} - \Delta_{Ps[k-1]}$ number of subperiods in the kth group can be used to consume $\Delta_{Ps[j]} - \Delta_{Ps[j-1]}$ surplus slots, with each one consuming one slot.*

Proof: The kth group contains at least $\Delta_{Ps[k]} - \Delta_{Ps[k-1]}$ number of subperiods can be identically written as:

$$Ps[k] - Ps[k-1] \geq \Delta_{Ps[k]} - \Delta_{Ps[k-1]} \tag{17}$$

Based on Eqn.(16), the right part of above inequality can be expressed as:

$$\Delta_{Ps[k]} - \Delta_{Ps[k]-1} = (Ps[k] - Ps[k-1]) \left(T_i^* - \sum_{\substack{j=1 \\ j \neq i}}^{N} \lfloor \frac{T_i^* x_j}{T_j} \rfloor - \min(\bar{x_i}) \right) \tag{18}$$

Then we have

$$\begin{aligned} & T_i^* - \sum_{\substack{j=1 \\ j \neq i}}^{N} \lfloor \frac{T_i^* x_j}{T_j} \rfloor - \min(\bar{x_i}) \leq 1 \Leftrightarrow \\ & \frac{H}{T_i^*} \cdot T_i^* - \frac{H}{T_i^*} \cdot \sum_{\substack{j=1 \\ j \neq i}}^{N} \lfloor \frac{T_i^* x_j}{T_j} \rfloor - \frac{H}{T_i^*} \cdot \min(\bar{x_i}) \leq \frac{H}{T_i^*} \Leftrightarrow \\ & H - \frac{H}{T_i^*} \cdot \min(\bar{x_i}) - H \cdot \sum_{\substack{j=1 \\ j \neq i}}^{N} \lfloor \frac{x_j}{T_j} \rfloor \leq \frac{H}{T_i^*} \Leftrightarrow \\ & N_s + H \cdot \sum_{\substack{j=1 \\ j \neq i}}^{N} \lfloor \frac{x_j}{T_j} \rfloor - H \cdot \sum_{\substack{j=1 \\ j \neq i}}^{N} \lfloor \frac{x_j}{T_j} \rfloor \leq \frac{H}{T_i^*} \Leftrightarrow N_s \leq \frac{H}{T_i^*} \end{aligned} \tag{19}$$

Thus, $Ps[k] - Ps[k-1] \geq \Delta_{Ps[k]} - \Delta_{Ps[k-1]}$. Moreover, as the actual usable surplus slots ($\Delta_{Ps[j]} - \Delta_{Ps[j-1]}$) within one group does not change, any of the $\Delta_{Ps[k]} - \Delta_{Ps[k-1]}$ subperiods within the group can be used to consume one of the $\Delta_{Ps[j]} - \Delta_{Ps[j-1]}$ surplus slots. □

Based on Lemma 6 and Lemma 7, we design Algorithm 2 to compute a realizable permutation of $\vec{x_i}$. Once the number of slots allocated to each subperiod of each device is determined, the slot-based transmission scheduling problem can be mapped to the task scheduling problem on a single processor, where the number of slots allocated for transmitting a packet corresponds to the CPU time for executing a task. Earliest Deadline First (EDF) policy has been proven to be an optimal scheduling algorithm on preemptive uniprocessors in the following sense: if a collection of independent jobs, each characterized by an arrival time, an execution requirement and a deadline, can be scheduled (by any algorithm) in a way that ensures all the jobs complete by their deadline, EDF will schedule this collection of jobs so they all complete by their deadline. Hence, the slot-based schedule can be computed using the EDF policy based on the allocation vector generated by Algorithm 2.

Algorithm 2: Compute a realizable permutation of $\bar{x}_i$

input : H,$x[i][m]$,T_i
output: Feasible order of $x[i][m]$

1 **if** *Exist* $\max(x[i][m]) \neq \min(x[i][m])$ *for any* d_i **then**
2 Calculate N_s according to (15);
3 Calculate Δ_l according to (16);
4 $\Delta_{min} = \Delta_{\frac{H}{T_i}}$; $N_g = 0$;
5 **for** $l = \frac{H}{T_i}$ *to* 1 **do**
6 **if** $\Delta_l < \Delta_{min}$ **then**
7 $\Delta_{min} = \Delta_l$;
8 N_g++;
9 Record the position of the last subperiod for the group G_{H/T_i^*-x} as $Ps[H/T_i^* - x] = l$;
10 The lth subperiod belong to the group G_{H/T_i^*-x};
11 **for** $j = 1$ *to* N_g **do**
12 $t = \Delta_{Ps[j]}$;
13 **for** $k = 1$ *to* t **do**
14 $x[i^*][Ps[j]]++$;
15 $Ps[j]--$;
16 $\Delta_{Ps[j+1]} -= \Delta_{Ps[j]}$;
17 Return right order of $x_{i,m}^*$;

VI. Performance Evaluation

In this section, we evaluate our scheme through extensive simulations in MATLAB. We compare our scheme (denoted as OPT-SLOT) with the Greedy Maximizer algorithm proposed in [8] in terms of maximizing the reward function and balancing award among different subperiods within each device. We also demonstrate the feasibility to provide a guarantee on reward by adjusting the weight for each device.

A. Comparison with Greedy Maximizer

In our scheme each device repeats the same schedule for each hyperperiod. Thus the rewards each device receives from the same subperiod in different hyperperiod are the same. However, the Greedy Maximizer algorithm maximizes the

system reward from a long-term average perspective, and the rewards each device receives from the same subperiod in different hyperperiod can be different. To make fair comparisons, we perform the simulations as follows: we choose seven network setups with different N and T_i, as described in Table II. For each network setup, we perform 1000 simulation runs with different setting on packet loss rate. In each simulation run, the packet loss rate for each link is chosen uniformly from the range of $[0.2, 0.8]$, and remains fixed during the simulation. Each simulation run lasts for 200 hyperperiods. In this set of simulations, the weights for all sensor devices in Eqn. (1) are set to 1, and the minimum reward requirement for each device is set to 0 in the Greey Maximizer algorithm.

TABLE II
NETWORK CONFIGURATIONS

Group_ID	N	T_i	Group_ID	N	T_i
$G1$	3	[3, 4, 6]	$G5$	4	[12, 5, 9, 6]
$G2$	3	[5, 7, 8]	$G6$	5	[6, 9, 12, 10, 8]
$G3$	4	[3, 4, 10, 7]	$G7$	5	[20, 15, 5, 6, 9]
$G4$	3	[20, 10, 7]			

Since the weights for all sensor devices are 1, $\mathcal{R}$ in Eqn. (1) is equivalent to the expected number of packets received within one hyperperiod, and the maximum reward equals to the total number of packets generated in one hyperperiod. We define the average reward in percentage as the ratio of the average reward in one hyperperiod to the maximum reward. Fig.3 compares the two schemes in terms of the average reward in percentage and its standard deviation in one hyperperiod. It can be seen that our scheme can successfully deliver 48% of the packets in the worst case and 95% of the packets in the best case. Greedy Maximizer delivers around 30% of the packets in most cases, and the performance for different network settings does not change too much. This is because Greedy Maximizer aims at improving performance by meeting the minimal reward requirement for each device instead of maximizing the reward. In each time slot, the task that has the maximum debt on reward requirement is given higher priority for scheduling, whereas our scheme focuses on maximizing the total reward in one hyperperiod.

Fig. 4 shows the average difference on the number of time slots allocated to different subperiods of the same device and its standard deviation. It can be seen that the average difference for our scheme is less than 0.2. This is because there is at most one device in which different subperiods can have different numbers of slots with maximum different of one (ref. Lemma 3), which indicates that each device can get almost the same reward in different subperiods. This intra-device fairness will improve the stability of the control systems. The transmission schedules constructed by the Greedy Maximizer algorithm have much bigger variations. As can be seen from Fig. 4, the average difference is around 2.5 and the maximum difference can be more than 8. This is because the system performance in Greedy Maximize is described by the long-term average reward, and in each slot the algorithm greedily chooses the

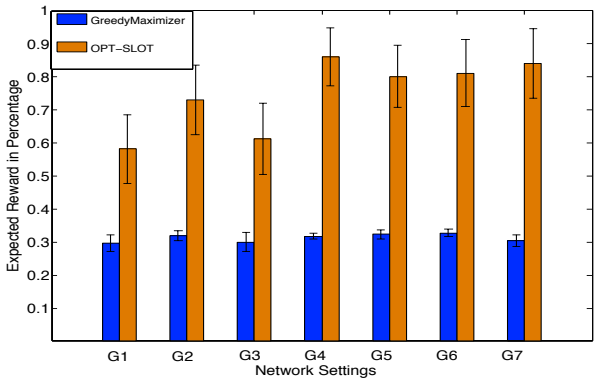


Fig. 3. Comparison on the expected reward under different network settings.

the task with the maximum debt on reward requirement for execution, thus leading to variations on the reward in different subperiods. These variations can affect the performance of the industrial systems.

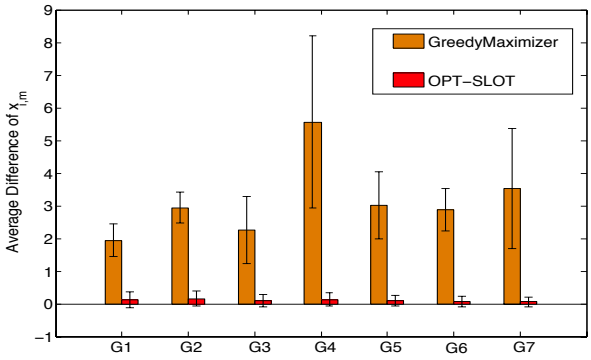


Fig. 4. Comparison the reward difference in different subperiods.

B. Impact of the Weight w_i on Performance

Our scheme provides fairness on reward in different subperiods of the same device, but does not guarantee fairness among different devices. A device with low packet loss rate can get high probability to be scheduled in our scheme. In many industrial control applications, each sensor device has a minimum requirement on packet delivery rate from itself to the controller. We use r_i to denote the minimum reward that should be obtained by d_i in a subperiod. Let x_i be the minimum number of time slots that should be allocated to d_i in each subperiod to meet the minimum requirement. We have

$$x_i = \lceil \frac{\log(1-r_i)}{\log p_i} \rceil. \tag{20}$$

For each d_i, there are $\frac{H}{T_i}$ packets generated within one hyperperiod. Then the following should be satisfied:

$$x_1 \cdot \frac{H}{T_1} + \cdots + x_N \cdot \frac{H}{T_N} \leq H \tag{21}$$

Bring (20) into (21), we can get

$$\sum_{i=1}^{N} \frac{1}{T_i} \cdot \lceil \frac{\log(1-r_i)}{\log p_i} \rceil \leq 1. \quad (22)$$

The above inequality enables to check the feasibility to meet the minimum requirements for a given network setting. Once the above inequality is satisfied, we can perform scheduling to meet the minimum requirements in the following two ways: (1) similar to the work in [3], the slots allocated to a packet is composed of two parts: the mandatory part (i.e. x_i) and the optional part. The mandatory part is firstly allocated to satisfy the minimum requirements, and then the optional part is allocated using our approach; (2) adjust the weight assigned to each device to meet the minimum requirements. As the first approach is straightforward, in the following we use an example to illustrate the second approach.

Fig. 5 shows the expected number of packets received by each device in a subperiod, where $N = 3$, $[T_1, T_2, T_3] = [3, 4, 5]$ and $[p_1, p_2, p_3] = [0.6, 0.8, 0.7]$. When all three devices have the same weight ($w_i = [0.33, 0.33, 0.33]$), d_1 has the maximum number of received packets in each subperiod. However, d_2 does not get any allocated slots due to its high loss probability. By changing the weight to $w_i = [0.25, 0.5, 0.25]$, d_2 gets more slots to be assigned and consequently the expected number of received packets in one subperiod is increased to 5.4, and the total expected number of packets received by the controller is 16.4. Even though the maximum reward decreases a little bit, all three devices have chance to deliver their packets to the controller. By carefully adjusting the weights, it is possible to construct a schedule that achieves the minimum requirement for each device. However, here we will not study the optimal solution for weight adjustment to meet the minimum requirement, which is left for future work.

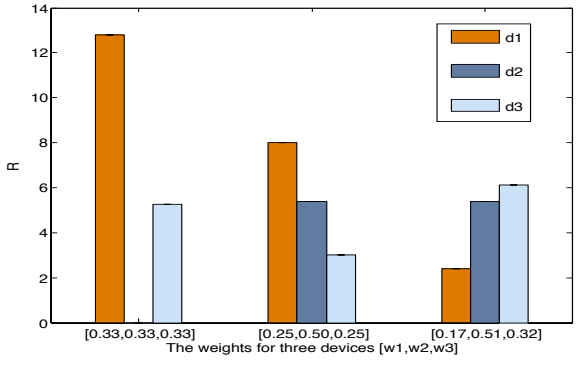


Fig. 5. Comparison of the expected number of packets received by each device in every subperiod under different settings of the weights.

VII. Conclusions

This paper investigates the problem of scheduling delay-constrained traffic in networks with unreliable wireless links. This category of problems is normally NP-hard especially when the traffic pattern is heterogeneous. We propose a two-stage scheduling approach in which slots are optimally allocated to subperiods of each device first to maximize the reward, and then the transmission schedule is constructed using slot-based scheduling. Even though the objective function is a nonlinear function of the number of slots allocated to each subperiod, which is commonly NP-hard to optimize, we propose algorithms with polynomial-time complexity to compute the optimal solution. Simulation results demonstrate that our scheme can yield much higher packet delivery rate than existing schemes. Moreover, our scheme ensures the fairness on reward for different subperiods of a device, which is important for maintaining the stability of control systems. Also we demonstrate that the reward for different devices can be controlled by adjusting the weights assigned to the devices to meet the minimal reward requirement for each device. Future work includes the analysis of the relationship between the weights and the minimal requirements on rewards.

References

[1] IEEE 802.15 WPAN^{TM} Task Group 4e (TG4e). In *http://www.ieee802.org/15/pub/TG4e.html*.

[2] V. Annamalai, S. Gupta, and L. Schwiebert. On tree-based convergecasting in wireless sensor networks. In *IEEE Wireless Communications and Networking (WCNC)*, pages 1942–1947, 2003.

[3] H. Aydin, R. Melhem, D. Mosse, and P. Mejía-Alvarez. Optimal reward-based scheduling for periodic real-time tasks. *IEEE Transactions on Computers*, 50(2):111–130, 2001.

[4] P. Djukic and S. Valaee. Distributed link scheduling for TDMA mesh networks. In *Proceedings of IEEE International Conference on Communications (ICC)*, pages 3823–3828, 2007.

[5] S. Ergen and P. Varaiya. TDMA scheduling algorithms for wireless sensor networks. *Wireless Networks*, 16(4):985–997, 2010.

[6] L. A. G. G. B. Francesco Capozzi, Giuseppe Piro and P. Camarda. Downlink packet scheduling in lte cellular networks: Key design issues and a survey. *IEEE Commun. Surveys and Tutorials*, pages 678–700, 2013.

[7] HARTCOMM. *WirelessHART specifications*. http://www.hartcomm2.org, 2007.

[8] I. Hou and P. Kumar. Scheduling periodic real-time tasks with heterogeneous reward requirements. In *IEEE 32nd Real-Time Systems Symposium (RTSS)*, pages 282–291, 2011.

[9] I.-H. Hou and P. Kumar. A survey of recent results on real-time wireless networks. In *The first workshop on real-time wireless for industrial applications (REALWIN)*, 2011.

[10] http://www.isa.org/. *ISA-100.11a-2009*, 2009.

[11] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. Real-time scheduling for wirelesshart networks. In *IEEE 31st Real-Time Systems Symposium (RTSS)*, pages 150–159, 2010.

[12] P. Soldati, H. Zhang, Z. Zou, and M. Johansson. Optimal routing and scheduling of deadline-constrained traffic over lossy networks. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–6, 2010.

[13] P.-J. Wan, S. C.-H. Huang, L. Wang, Z. Wan, and X. Jia. Minimum-latency aggregation scheduling in multihop wireless networks. In *MobiHoc '09: Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, pages 185–194, 2009.

[14] Y. Wang, W. Wang, X.-Y. Li, and W.-Z. Song. Interference-aware joint routing and tdma link scheduling for static wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, pages 1709–1726, 2008.

[15] A. Willig. Recent and emerging topics in wireless industrial communications: A selection. *IEEE Transactions on Industrial Informatics*, 4(2):102 – 124, 2008.

[16] H. Zhang, M. Albert, and A. Willig. Combining TDMA with slotted aloha for delay constrained traffic over lossy links. In *proceeding of the 12th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2012.